DOE/ER/03077-206

Courant Mathematics and
Computing Laboratory

U.S. Department of Energy

Evaluation of Step Directions in Optimization Algorithms

William C. Davidon and Jorge Nocedal

Research and Development Report

Prepared under Contract DE-AC02-76ER03077
with the U.S. Department of Energy, Division of Basic
Energy Sciences, Applied Mathematical Sciences Program

Mathematics and Computers

December 1983

NEW YORK UNIVERSITY

DOE/ER/03077-206
UC-32
Mathematics and Computers

Courant Mathematics and Computing Laboratory

New York University

EVALUATION OF STEP DIRECTIONS IN OPTIMIZATION ALGORITHMS

William C. Davidon* and Jorge Nocedal**

December 1983

*Department of Mathematics, Haverford College, Haverford, PA 19041. This work was done while this author was visiting the Courant Institute of Mathematical Sciences

**Courant Institute of Mathematical Sciences, New York University, New York, NY 10012. Present address: EE/CS Dept., Northwestern University, Evanston, IL 60201.

Prepared under Contract DE-AC02-76ER03077
with the U. S. Department of Energy.

DISCLAIMER

Printed in U.S.A.

Available from

National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

TABLE OF CONTENTS

ABSTRACT

We present a method for comparing that part of optimization methods which chooses each step direction. It is an example of a general approach to algorithm evaluation in which one tests specific parts of the algorithm, rather than making overall evaluations on a set of standard test problems. Our testing procedure can be useful for developing new algorithms and for writing and evaluating optimization software.

We use the method to compare two versions of the conjugate gradient algorithm, and to compare these with an algorithm based on conic functions.

Key words and Phrases: optimization, software evaluation, conic functions, conjugate gradients

1. Introduction

A common procedure for comparing optimization algorithms consists of testing them on a standard set of problems. An evaluation is based on criteria such as the total number of function evaluations and the average number of successful runs; see for example Moré, Garbow and Hillstrom (1981). If these tests are carefully prepared they will often give a good indication of the efficiency, reliability and robustness of optimization software. Nevertheless, this overall evaluation sometimes does not provide enough detailed information to help us determine the strengths and weaknesses of an algorithm. A program could perform poorly for very different reasons. For example, the line search could be inadequate, in which case a simple modification could solve the difficulties; or the search directions could be poor, in which case there could be no cure. From these types of tests it may be very difficult to determine where the deficiencies are; see for example Hiebert (1981).

In this paper we present a method that tests a specific and very important part of optimization algorithms, namely the choice of the step directions. For this we study a single iteration of each algorithm in an environment that emphasizes its essential character and minimizes the importance of programming decisions. The method is invariant under affine transformations of both the domain and range of the objective function. It is used to compare two versions of the conjugate gradient algorithm, and to compare these with an algorithm based on conic models. We study only algorithms that use line searches; however, we believe that this approach could be useful for analyzing trust region methods as well.

The motivation for this work came from the development of an algorithm that uses conic models. There are many possible implementations of this algorithm, and some computational experience was desired. Overall evaluations on standard

test problems could not be used, since there were too many unknowns. By using the evaluation method described in this paper, we have been able to observe the performance of the various conic models. We have also reexamined the conjugate gradient algorithm and found that the choice of the two points used by the algorithm in generating a new search direction is important.

We believe that the major application of this approach will be in the development of new algorithms and the writing of their software. The efficiency of optimization codes is very dependent on various decision parameters, such as the accuracy of the line search, the restart tolerance, or the trust region size. A good code is developed by choosing these parameters wisely, and its success depends largely on them. When a new type of algorithm is being developed, it is difficult to know how to choose the parameters; a great deal of numerical experimentation might be required. It would be preferable to find a way of isolating parts of the algorithm in order to gain useful information about each of the parameters. In this paper we present an example of how to do this. In Section 2 we outline a procedure for deciding how good a search direction is. We then show, in sections 3 and 4, how one can vary some decision parameters independently and observe their effect on the search direction. We study here only the conjugate gradient method and a conic method. Other optimization algorithms require somewhat different procedures; however, the main idea remains the same.

By making a careful analysis of search directions Powell (1977) was able to explain why under some circumstances the Fletcher-Reeves method does not perform as well as the Polak-Ribiere method. The basis of his analysis is very similar in nature to the method presented here, and is evidence of the fruitfulness of this approach.

2. Evaluation of a Step Direction

Our goal is to develop a procedure for deciding how useful a direction is. The idea is to define an initial direction, let the algorithm choose a new search direction, and by exploring the function along it, determine how good the new direction is. It is important that the procedure be invariant under affine transformations of both the domain and range of the objective function. We propose, among several possible procedures, the following:

1. Choose an initial point x_0 and an initial direction d_0.
2. Apply the algorithm and generate a new search direction d_i. (Call x_i the point where d_0 and d_i intersect, i.e., the starting point of the new iteration).
3. Locate the minimizer x_* of the function along d_i. Compute the relative decrease in f:

$$r = \frac{f_i - f^*}{f_o - f_i}, \qquad (1)$$

where f_*, f_o and f_i are the function values at x_*, x_o and x_i.

The quotient r will be the measure of the quality of a search direction. We assume that, in the region of interest, there is a unique one-dimensional minimizer of the function along d_i. Note that some algorithms may require additional information in step 2. For example, variable metric methods require a matrix that is updated during the iteration.

To apply this procedure we start by choosing a test function, for example one with singularities or narrow curving valleys. Next we select the starting point and direction in a region of interest of the function. We then apply the various algorithms to be tested and record how good their search directions are. We will now describe in detail two applications of this procedure.

3. Application to two versions of the conjugate gradient algorithm

The conjugate gradient algorithm generates a search direction based on information about the function at two points; see for example Fletcher and Reeves (1964). It is common to use for this the initial and final points in the line search. As the starting point is fixed, the only variable in the algorithm is the second point. In other words, the only free parameter of the algorithm is the accuracy in the line search. We will not follow this practice, but will allow more freedom in choosing the two points required by the algorithm. During the line search, several intermediate points are computed and some of these can be considered as candidates.

To study the effect of the choice of the two points on the search directions we will consider two strategies:

CG(1) We use the starting point, which is given, and the one-dimensional minimizer of the function along the initial direction.

CG(2) We also find the one-dimensional minimizer, which will be one of the points used. The second point is either the starting point or the first trial point in the line search, whichever gives the lowest function value. The choice of the trial point will be arbitrary.

We will now comment on these strategies. The first one resembles the common use of the conjugate gradient algorithm, except for the use of an accurate one-dimensional minimizer. This can be changed to find only a rough estimate of the minimizer, but this introduces some complications, as will be discussed later.

The second strategy is based on the observation that even though the conjugate gradient algorithm requires only two data points to generate a new search direction, it should use at least three points during the line search. This is because we want to ensure that the line search performs at least one

quadratic fit, so that if the objective function is quadratic the algorithm will maintain conjugacy. Therefore for every search direction the method will generate at least three points: the starting point, the trial point and an estimate of the minimizer. It may generate other intermediate points depending on the objective function, the type of line search, and the accuracy of the search. It is not desirable to consider the intermediate points, because they depend on the details of the line search. We therefore opted for considering only three candidate points: the starting point and the minimizer (which are uniquely specified) and the trial point which will be considered as the variable in our tests. The reason that the trial step is not fixed is that there is no accepted way of choosing it. Various suggestions have been implemented in the modern codes, but they are to some extent arbitrary. In the numerical tests described below we will vary the value of the trial point within a prescribed interval around the minimizer and will observe its effect in the generation of search directions.

The following results were obtained using double precision arithmetic on a VAX 11/780 at the Courant Mathematics and Computing Laboratory. The starting point is denoted by x_o, the initial direction by d_o and the minimum of the function along d_o by f_1. The steplength to the minimizer along the initial direction is denoted by λ, and the minimum value of the function along the new search direction by f_*. The relative decrease in the function, denoted by r, is given by (1). The test functions are taken from More', Garbow and Hillstrom (1981).

Problem 1 Rosenbrock's function, $x_o = (0,0)^t$, $d_o = (1,0)^t$

$\lambda = .161$, $f_1 = .771$, $f_o = 1.00$

For CG(1): $f_* = .624$, $r = .642$

For CG(2):

TRIAL STEPLENGTH

	.01	.05	.167	.3	.5	.1
f*	.626	.615	.474	.624	.624	.624
r	.633	.681	1.30	.642	.642	.642

Problem 2 Rosenbrock's function, $x_o = (0,0)^t$, $d_o = (.1,-.015)^t$

$\lambda = .102$, $f_i = .872$, $f_o = 1.00$

For CG(1): $f_* = .729$, $r = 1.12$

For CG(2):

TRIAL STEPLENGTH

	.01	.05	.1	.2	.6	2.0
f*	.719	.670	.588	.729	.729	.729
r	1.19	1.58	2.22	1.12	1.12	1.12

Problem 3 Powell's singular function, $x_o = (2,-.9,.2,-.2)^t$

$d_o = (-1.7,.8,-.1,.7)^t$, $\lambda = 2.08$, $f_i = 1.21$, $f_o = 287$

For CG(1) $f_* = .819$, $r = .0014$

For CG(2):

TRIAL STEPLENGTH

	1.0	1.5	2.0	3.0	6.0	15.
f*	.630	1.99	.255	.783	.819	.819
r	.0020	.0035	.0033	.0015	.0014	.0014

We can see from these results that the second strategy, CG(2), is usually better. When the trial point is near the one-dimensional minimizer, CG(2)

generates a much better search direction than CG(1).

We present only a small sample of results, but it is representative of what we have observed. The purpose of this paper is to describe an approach for evaluating algorithms; therefore we will not make extensive comparisons of different methods. Note that there is very little extra computational cost in CG(2) and that this type of strategy is not difficult to implement. Therefore we suggest that it should be tried in the various conjugate gradient-type codes currently in use. There are several precautions one should take while doing this. The two points may be placed so that large numerical errors are introduced while computing the new search direction. One should take this into account before deciding which two points to use. Also, the condition $s^t y>0$ should be enforced if the algorithm performs a variable metric update (here s represents the difference between the two points used and y the difference in gradients). Examples of these methods are the code of Shanno and Phua (1980) and variable storage methods.

We should point out that, even though in the results presented above the second strategy is always preferable, there are cases where it is not. However this occurs rarely and in the majority of the tests CG(2) is superior.

The evaluation method just described makes use of accurate one-dimensional minimizers. This simplifies the testing but does not reflect the practical use of optimization algorithms. This can be changed and rough estimates of the minimizer can be used. However, one should make sure that the choice of this estimate of the minimizer is the same for all algorithms tested, and if possible, that this choice is invariant under affine transformations of both the domain and range of the function.

Overall our testing procedure suggests that the selection of the two points can be important in the conjugate gradient method, and that the common practice of performing the simplest possible line search may not be justified.

4. Application to an algorithm based on conic functions

We will now consider our testing procedure to evaluate a new optimization algorithm, that will be called the conic algorithm. It is an extension to general functions of the methods described by Davidon(1982) and Gourgeon and Nocedal(1982). To generate a new search direction the conic algorithm requires function information at three points. These must satisfy certain conditions to ensure the existence of a normal conic fit. In the tests described below the three points are the starting point, the one-dimensional minimizer and the first trial point in the line search. As in the case of the conjugate gradient method the trial point will be allowed to vary, but must be such that a normal conic fit is possible.

We will choose an initial point and direction, and will study the effect of the trial point on the new search direction. This algorithm will be compared with CG(2), the better of the two conjugate gradient methods. The conic algorithm used in these tests has approximately the same storage and computational requirements as the conjugate gradient method; therefore we are comparing similar algorithms.

Here are some results on the same functions tested in the previous section. The problem numbers correspond to those of section 3.

Problem 1

TRIAL STEPLENGTH		.01	.05	.167	.3	.5	1.0
CONIC	f*	.546	.487	.340	.332	.447	.556
	r	.983	1.24	1.88	1.92	1.41	.939
CG(2)	f*	.626	.615	.474	.624	.624	.624
	r	.633	.681	1.30	.642	.642	.642

Problem 2

TRIAL STEPLENGTH		.01	.05	.1	.2	.6	2.0
CONIC	f*	.466	.499	.452	.504	.654	.710
	r	3.17	3.30	3.28	2.87	1.70	1.27
CG(2)	f*	.719	.670	.588	.729	.729	.729
	r	1.19	1.58	2.22	1.12	1.12	1.12

Problem 3

TRIAL STEPLENGTH		.1	.5	1.0	1.5	2.0	3.0
CONIC	f*	.678	.606	.377	.251	.261	.790
	r	.0019	.0021	.0030	.0033	.0033	.00147
CG(2)	f*	.813	.774	.630	.199	.255	.783
	r	.0014	.0015	.0020	.0035	.0033	.0015

In these tests the conic algorithm compares favorable with the best strategy for the conjugate gradient method, and is definitely more successful than the usual conjugate gradient method CG(1). Again, we only present a small set of results and leave detailed comparisons for a future study.

The main conclusion drawn from these tests is that conic fits are much more stable than anticipated. The practical implementation of conic methods has been slowed down by the lack of clear criteria for choosing three points that will generate a good conic models. It was feared that different choices could give drastically different results, because conic functions vary rapidly in magnitude near a singular hyperplane. The tests have shown that as long as the three points are not too widely separated, the conic models will produce good estimates of the minimizers. This can be done by safeguarding the line search using one of the usual controls. Note that the conic algorithm is not much more sensitive than the conjugate gradient method to the choice of the trial steplength. Several other aspects remain to be studied before conic algorithms can be implemented in general purpose codes. The evaluation procedure described in this paper will be useful for these studies. Up to now it suggests that conic models can make a contribution in the development of algorithms for nonlinear optimization.

References

1. Davidon, W.C. (1982). Conjugate directions for conic functions, in M.J.D. Powell, ed., Nonlinear Optimization 1981, Academic Press.

2. Fletcher, R. and Reeves, C. (1964). Function minimization by conjugate gradients, Computer Journal 7, 149-154.

3. Gourgeon, H. and Nocedal, J. (1982). A conic algorithm for optimization, Tech. Rep. Courant Institute, New York University, to appear in SIAM J. Scientific Stat. Computing.

4. Hiebert, K. L. (1981). An evaluation of mathematical software that solves nonlinear least squares problems, ACM Trans. Math. Software 7, 1-16.

5. Moré, J. J., Garbow, B. S. and Hillstrom, K. E. (1981). Testing unconstrained optimization software, ACM Trans. Math. Software 7, 17-41.

6. Powell, M.J.D. (1977). Restart procedures for the conjugate gradient method, Math. Programming 12, 241-254.

7. Shanno, D. and Phua, K. (1980). Remark on Algorithm 500, ACM Trans. Math. Software 6,4. pp. 618-622.